UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Circuiti in logica sequenziale

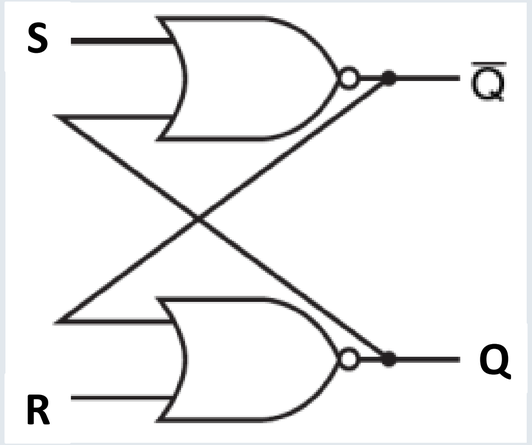
Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 25 marzo 2026

RECAP

S-R Latch è un circuito composto da 2 porte NOR concatenate. S = Set e R = Reset.



Input S	Input R	Stato Interno Old Q	Output Q	Output $\overline{Q}$
0	0	0	0	1
0	0	1	1	0
0	1	1/0	0	1
1	0	1/0	1	0
1	1	1/0	0	0

S = 0 e R = 0
Configurazione di riposo

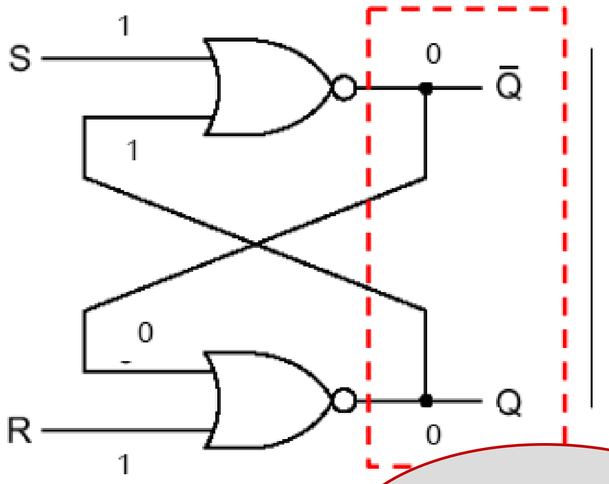
Input S	Input R	Stato Interno Old Q	Output Q	Output $\overline{Q}$
0	0	0	0	1
0	0	1	1	0
0	1	1/0	0	1
1	0	1/0	1	0
1	1	1/0	0	0

S = 1 e R = 0
Configurazione di SET

Input S	Input R	Stato Interno Old Q	Output Q	Output $\overline{Q}$
0	0	0	0	1
0	0	1	1	0
0	1	1/0	0	1
1	0	1/0	1	0
1	1	1/0	0	0

S = 0 e R = 1
Configurazione di RESET

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0



Input S	Input R	Stato Interno Old Q	Output Q	Output $\overline{Q}$
0	0	0	0	1
0	0	1	1	0
0	1	1/0	0	1
1	0	1/0	1	0
1	1	1/0	0	0

Potenziale instabilità

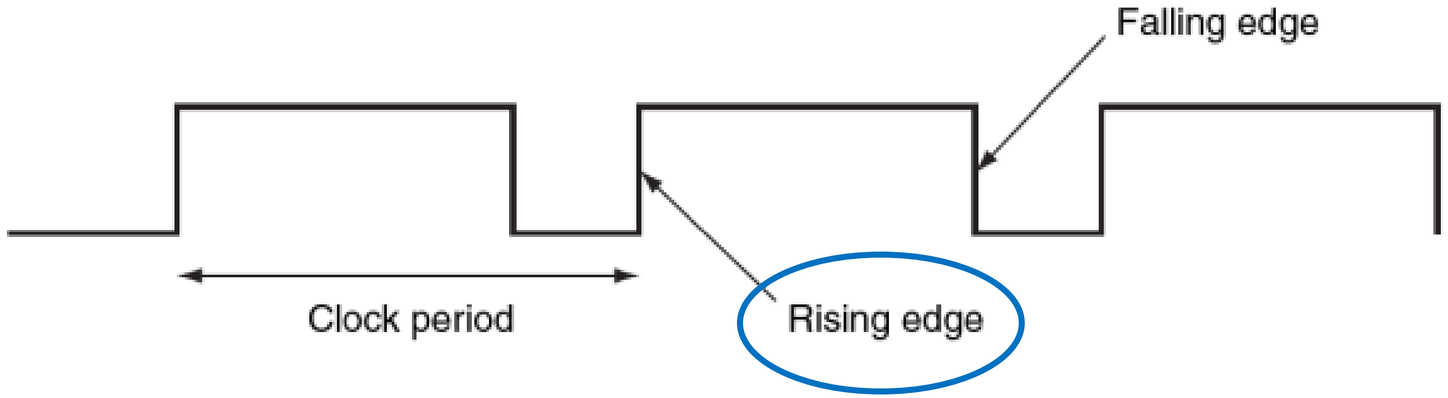
S	R	Stato
0	0	Old Q
1	0	1
0	**1**	0
1	1	-

UNIVERSITÀ DEGLI STUDI DI TRIESTE

SOLUZIONE: RENDERE IL CIRCUITO SINCRONO

Il segnale di clock è fondamentale per le reti sequenziali che sono caratterizzate da uno stato

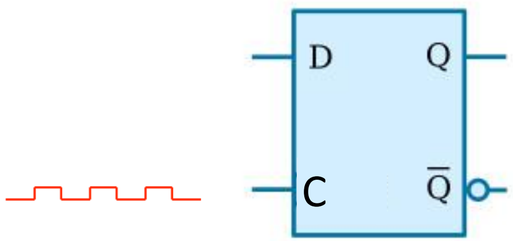
Clock – definito come un segnale (onda quadra) con un periodo predeterminato e costante.



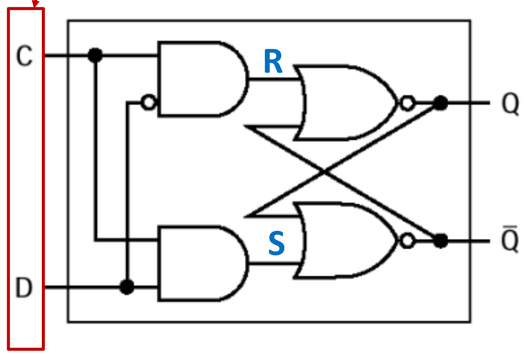
Caratterizzato da un periodo T (oppure ciclo) di clock e dalla frequenza F(definita come 1/periodo) e misurata in Hertz F= 1/T. **Il periodo è definito come il tempo tra due fronti di salita.**

Nota. Il periodo in cui il clock è a 1 potrebbe durare diversamente dal tempo in cui sta a 0.
Es: con **duty cycle** di 50% i due tempi sono uguali.
Se la circuiteria è «fatta bene» il *duty cycle* non è un parametro critico. ***Quello che conta è T!***

D LATCH



Errata!



Diventa quindi *impossibile* avere S=R=1.

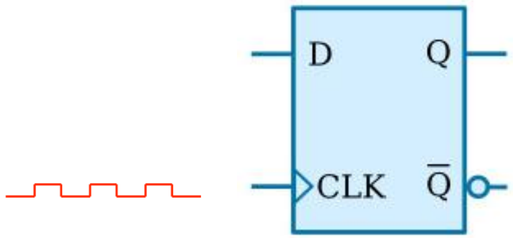
Nota: il D latch è un circuito *level-sensitive*, ovvero si attiva (aggiorna il suo output) ogni volta che il clock è alto.

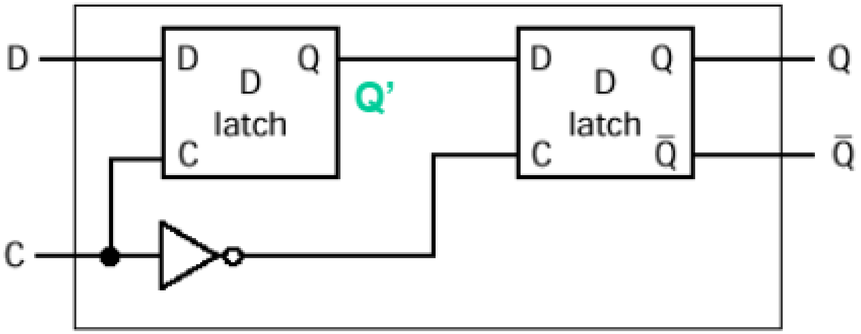
CLOCK	D	Q(T)	Q(T+1)	
0	-	0	0	Configurazione di riposo
0	-	1	1	
1	0	-	0	Reset
1	1	-	1	Set

Q(T): stato precedente; Q(T+1): stato successivo

!Trasparenza indesiderata: mentre il clock è alto, l'ingresso si propaga direttamente all'uscita, quindi Q potrebbe cambiare più volte nello stesso ciclo.

D FLIP FLOP



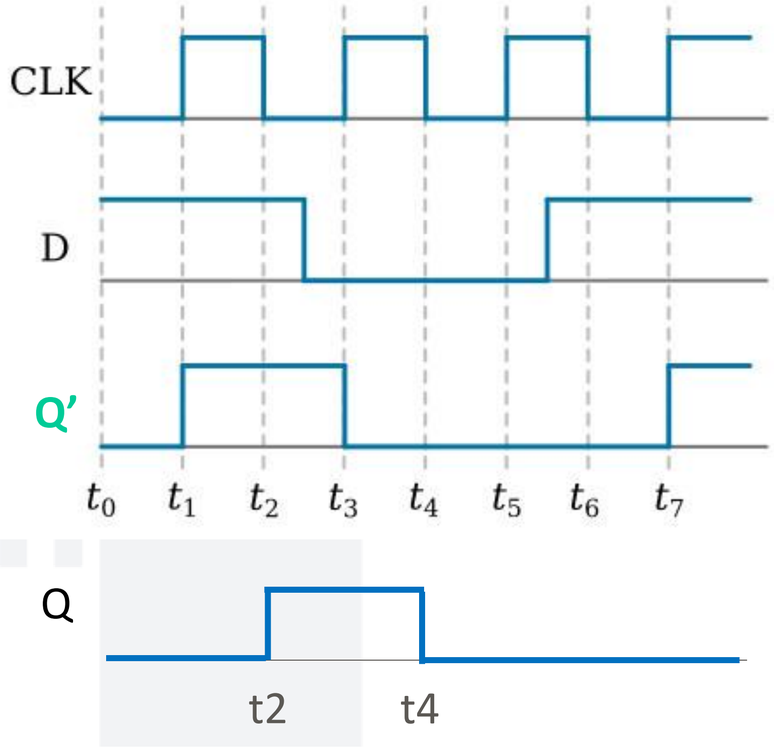


Clock	D	Q(t)	Q(t+1)
Fronte di salita	0	-	0
Fronte di salita	1	-	1
0	-	0	0
0	-	1	1
1	-	0	0
1	-	1	1

-: irrilevante

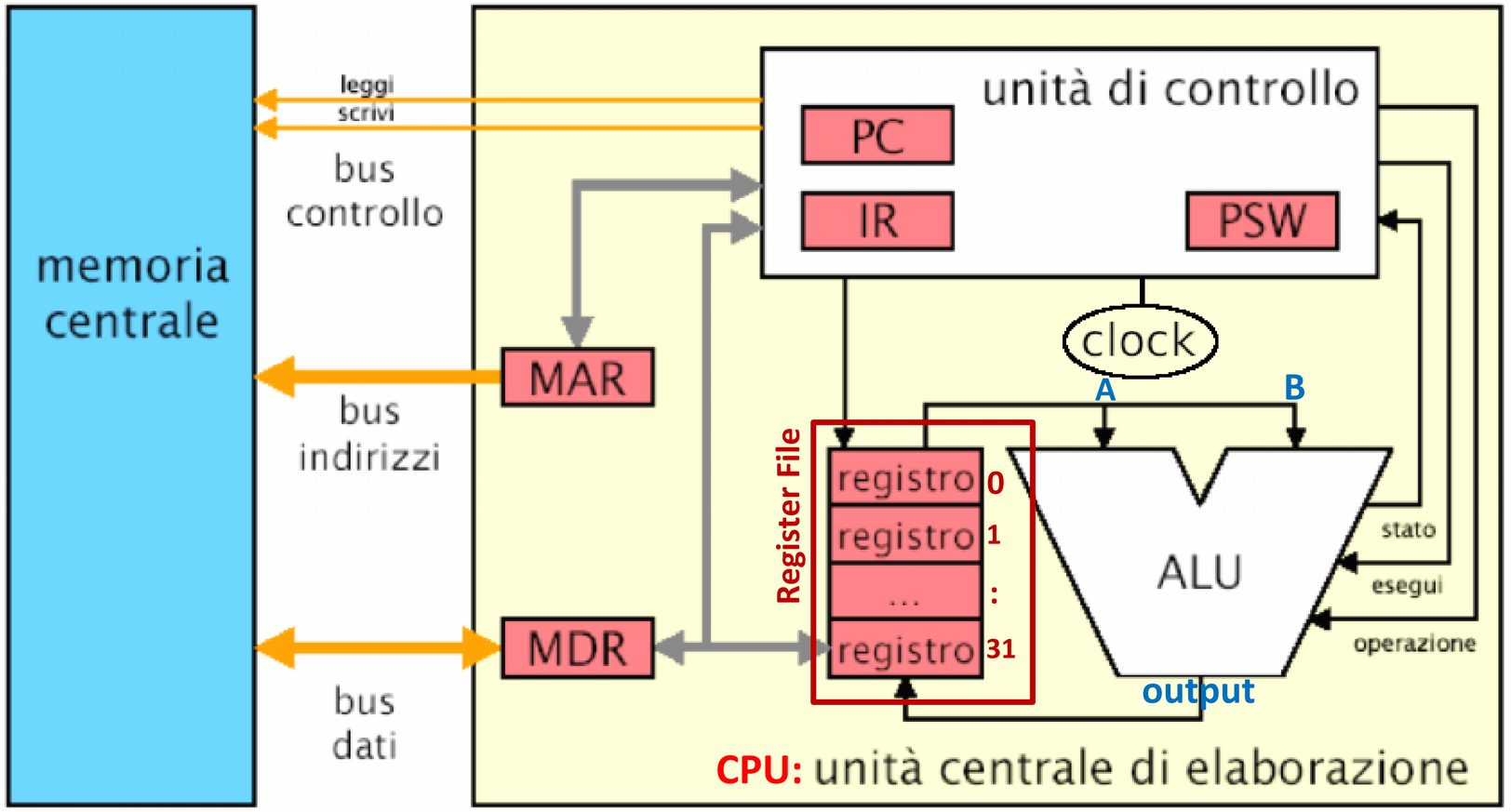
Esempio:

Nota: sono visualizzati solo alcuni istanti di tempo



UNIVERSITÀ DEGLI STUDI DI TRIESTE

RECAP



IR: Instruction Register

→ È **l'istruzione** che la CPU sta *decodificando/eseguendo*.

PC: Program Counter

→ Contiene **l'indirizzo** della prossima istruzione.

MAR: Memory Address Register

→ Contiene l'indirizzo di memoria da cui leggere o in cui scrivere, va sul bus indirizzi.

MDR: Memory Data Register

→ Contiene il dato letto dalla memoria o da scrivere in memoria, va su bus dati.

PSW: Program Status Word

→ Serve per decisioni di controllo (branch, condizioni). Il registro di stato: contiene flag (zero, carry, overflow, segno, ecc.).

DOVE CI SERVE LA LOGICA SEQUENZIALE?

DOVE CI SERVE LOGICA SEQUENZIALE?

Usano **logica sequenziale**:

- Register File
- Instruction Register (IR)
- MAR (Memory Address Register)
- MDR (Memory Data Register)
- Memoria (RAM/Cache)

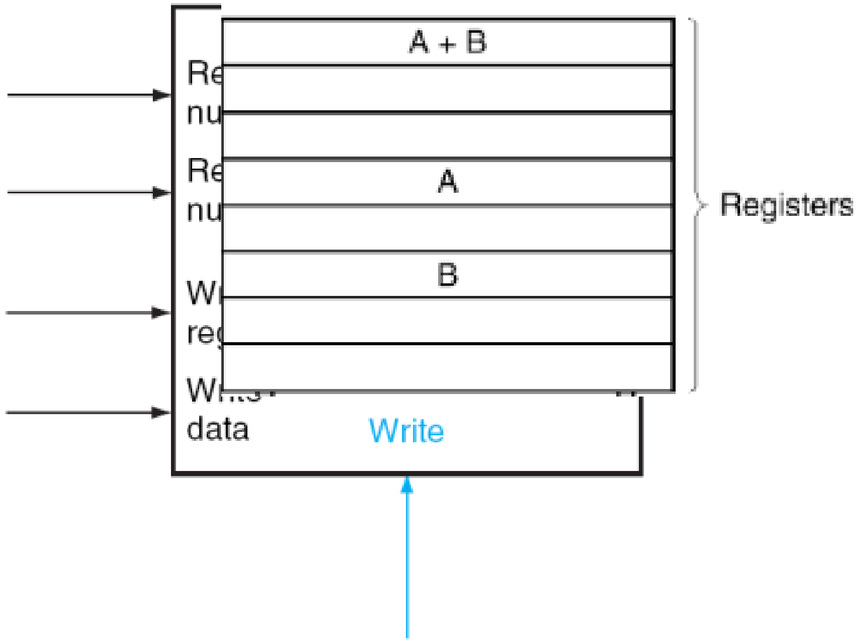
usano D flip flop

Nota: per la RAM si usa tecnologia diversa perché più conveniente (es. transistor e condensatori)

In generale: tutto ciò che **mantiene uno stato nel tempo**

REGISTRI E REGISTER FILE

Register File, Instruction Register (IR), Memory Address Register (MAR), Memory Data Register (MDR) hanno in comune il fatto di essere **REGISTRI**.



Un registro è costituito da n flip-flop

- Nel MIPS ogni registro è di 1 word = 4 byte = 32 bit

I registri sono organizzati in un Register File

- Il Register File del MIPS ha 32 registri (32 x 32 = 1024 flip flop)
- Il Register File permette la lettura di 2 registri e la scrittura di 1 registro

REGISTER FILE

Perché 5 bit?
Dove li trovo?

Read Reg1 # (5 bit)
numero del 1o registro da leggere

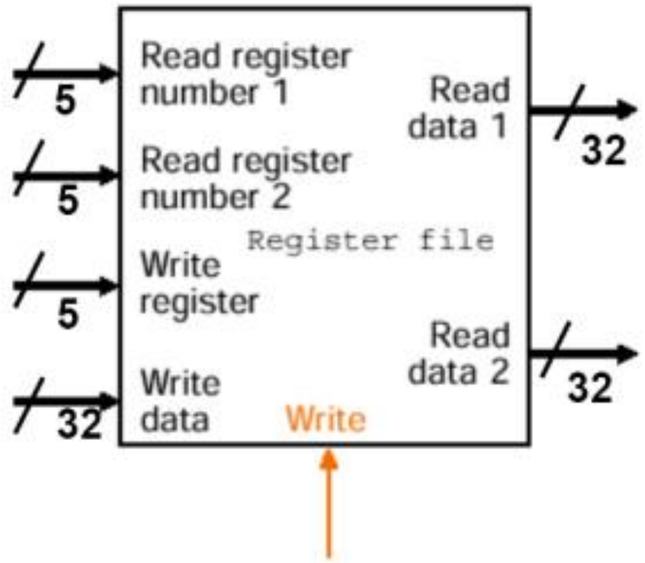
Read Reg2 # (5 bit)
numero del 2o registro da leggere

Read data 1 (32 bit)
valore del 1o registro, letto sulla base di Read Reg1 #

Read data 2 (32 bit)
valore del 2o registro, letto sulla base di Read Reg2 #

Write Reg # (5 bit)
numero del registro da scrivere

Write data (32 bit)
valore da scrivere nel registro Write Reg #



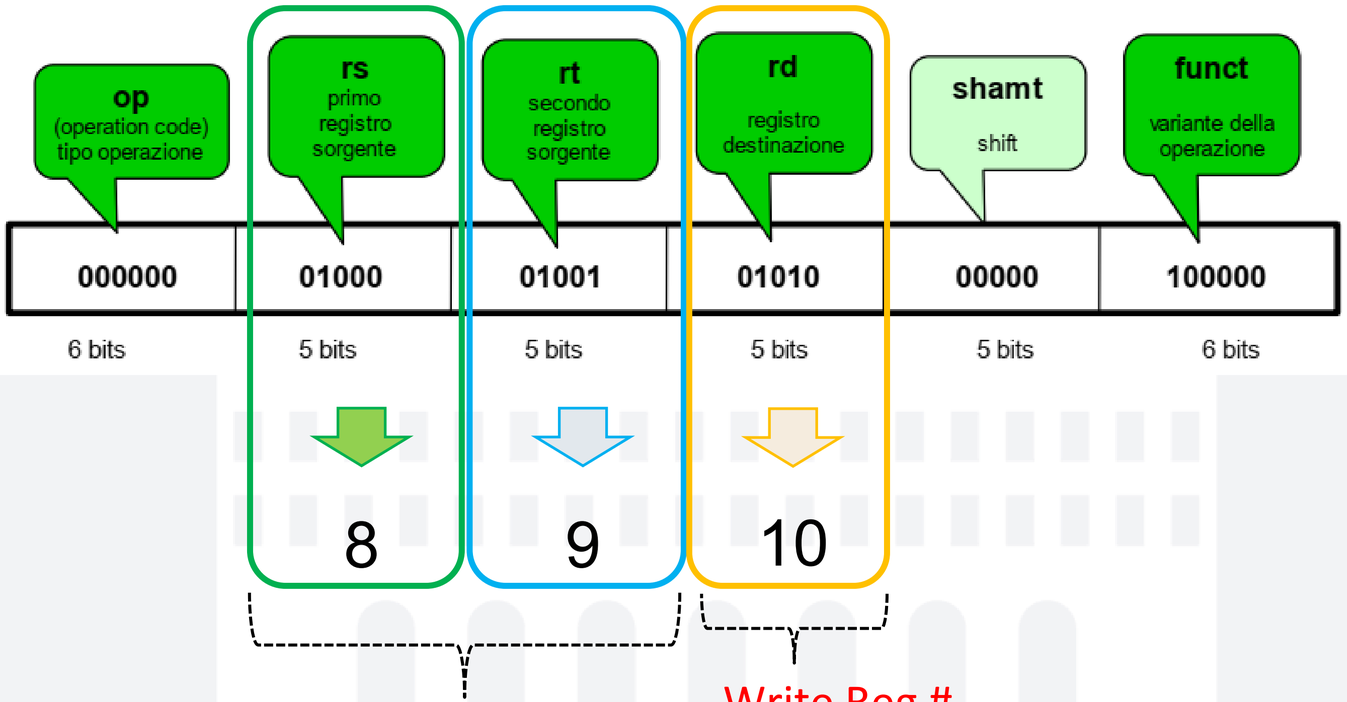
Write

- segnale di controllo messo in AND con il *clock*
- solo se Write=1, il valore di Write data viene scritto in uno dei registri

RICORDATE LA CODIFICA ISA DI UN'ISTRUZIONE?

00000001000010010101000000100000

add $10, $8, $9



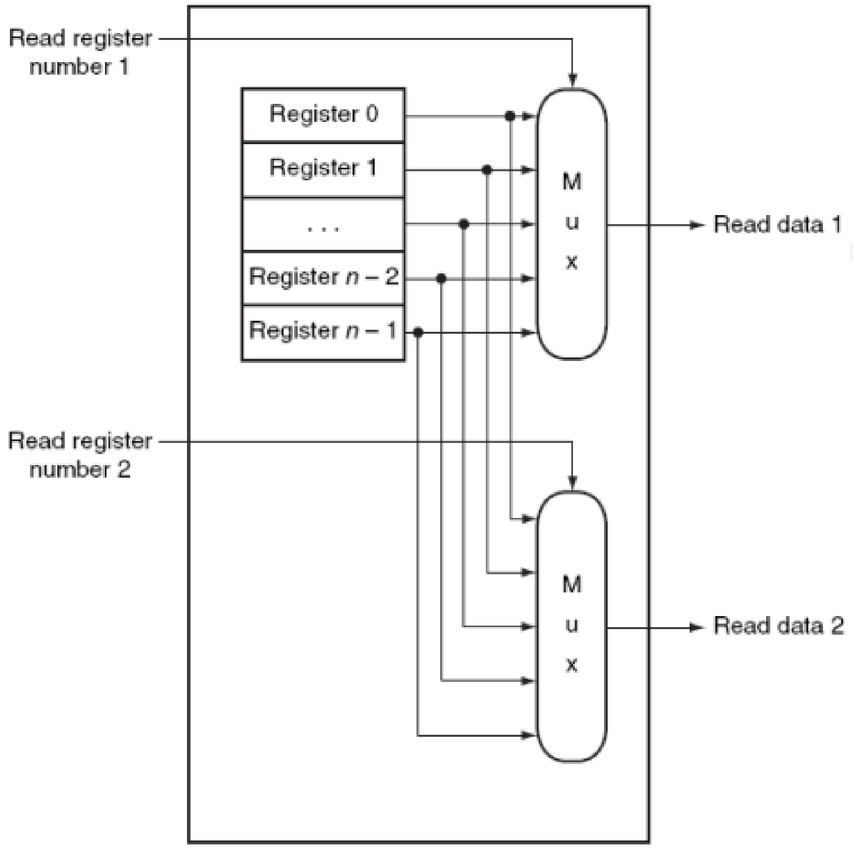
E adesso come prelevo il dato nel registro selezionato?

LETTURA DAL REGISTER FILE

- Utilizza i **2 segnali** che indicano i registri da leggere (Read Reg1, Read Reg2)

- Utilizza **2 multiplexer***: ognuno con 32 ingressi e un segnale di controllo da 5 bit

- Il Register File fornisce sempre **in output una coppia di registri**



**Nota*: *visualizzazione compatta: in realtà ogni mux è la serie di 32 mux dove entrano 32 linee da 1 bit*

SCRITTURA NEL REGISTER FILE

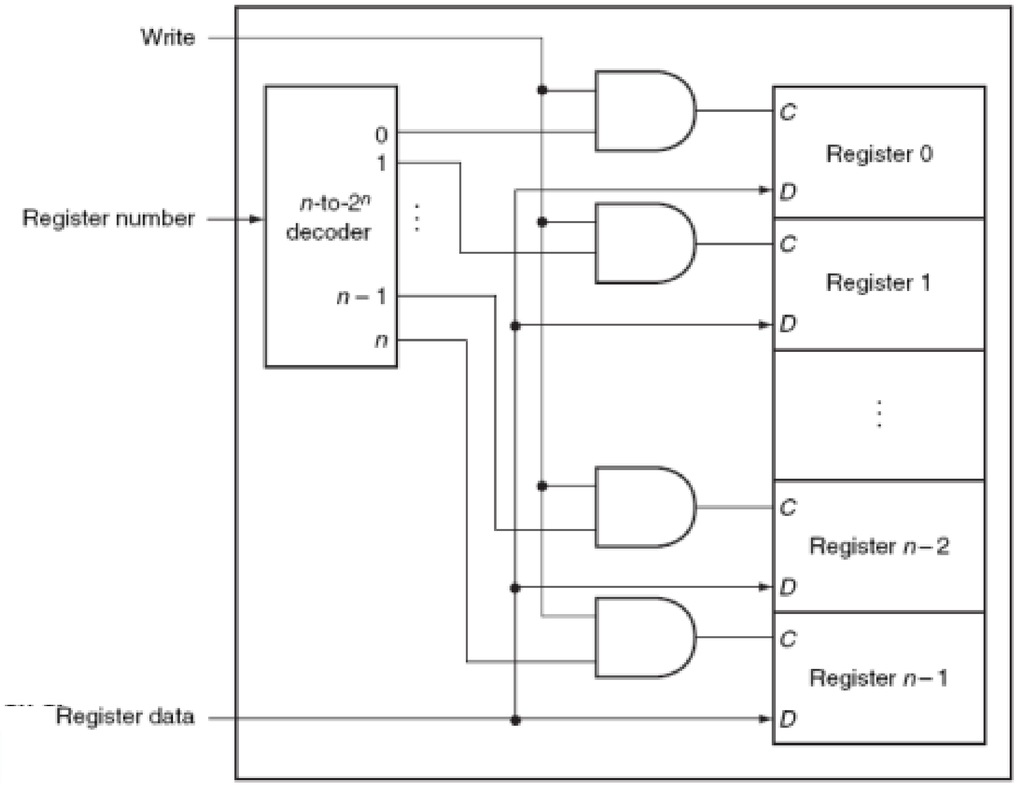
Utilizza **3 segnali di input/controllo**:

- Il registro da scrivere (Write Reg Number)
- Il valore da scrivere (Register Data)
- *Il segnale di controllo (Write)*

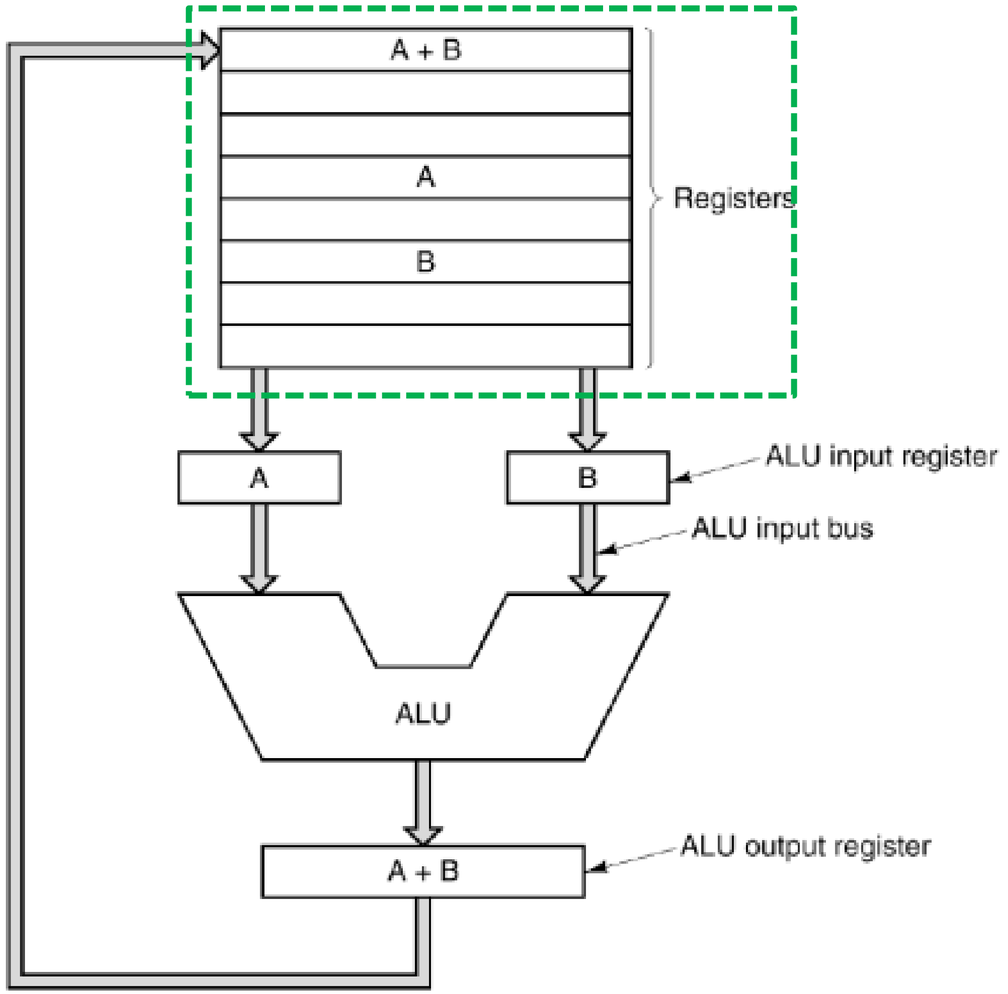
Utilizza **un decoder** che decodifica il numero del regis da scrivere (Write Register).

Il segnale Write (già in AND con il clock) è in AND con l'output del decoder.

Se Write non è *asserted dal clock* nessun valore sarà scritto nel registro.



INTERFACCIA REGISTER FILE CON ALU



Materiale per la lezione

- Appendice B Patterson & Hennessy